A COMPARATIVE STUDY OF APACHE BEAM RUNTIMES: APACHE SPARK AND APACHE FLINK

DATA INTENSIVE COMPUTING
PROJECT REPORT
2019

AUTHORS:

M Haseeb Asif <mhasif@kth.se>
Sruthi Sree Kumar<sruthisk@kth.se>

INTRODUCTION

Big data is growing very rapidly. With the increase in the amount of data, different data processing engines are being developed to meet the industry needs. Most of the platform surpass each other in different types of data and processing models. Apache beam is a project that let you define a model to represent and transform datasets irrespective of any specific data processing platform. Once defined, you can run it on any of the supported run-time frameworks which includes Apache Apex, Apache Flink, Apache Spark, and Google Cloud Dataflow. In this project, we are doing a performance comparison study of two runtimes for the Apache beam, Apache Spark and Apache Flink. We will look at the processing time for each runtime for a specific dataset.

EXPERIMENT METHODOLOGY

We used two different pipelines to compare the performance of different runners within Beam. First we have used one of the word count example from Apache Beam which loads the shakespeare literature and runs the word count on it. Secondly we have written a pipeline which will process the stored tweets. We are comparing the execution time for each of the different runners.

We collected the tweets data for the Diwali tweets, since it was trending on twitter. We used Twarc utility, an open source utility to fetch the twitter data. Twarc is a command line tool and Python library for archiving Twitter JSON data. Each tweet is represented as a JSON object that is exactly what was returned from the Twitter API. Tweets are stored as line-oriented JSON. Twarc will handle Twitter API's rate limits for you. In addition to letting you collect tweets Twarc can also help you collect users, trends and hydrate tweet ids.

We installed Twarc using pip command. Then we configured the twitter API secret key for authorization. Once configured it will use the twitter API key and run the search given the keyword and we are storing the result into a file using JsonL format. JsonL store each json item per line which is easy to read for further processing. The steps which we ran are listed below.

```
pip install twarc # installed
twarc configure # configure twitter API key
twarc search diwali > tweets.jsonl #get tweets related to the diwali and store
```

We have used HP probook 450 G5 with Intel Core i5-8250U having 8 GB of RAM to run the experiments. We ran both of the pipelines multiple times using Beam direct runner, spark runner and flink runner. We took observations of the execution time for each iteration.

PROJECT SETUP & EXECUTION

Setup of Beam Development environment to work with Beam's Java SDK.

1. Download and install JDK 8 (if its not installed already) and set the JAVA_HOME using
2. Download and install Apache Maven (3.6.2 is used here) and set the Maven Path using
3. Create a Maven Project and add the Beam dependency.

```
<dependency>
  <groupId>org.apache.beam</groupId>
  <artifactId>beam-sdks-java-core</artifactId>
  <version>[2.16.0, 2.99)</version>
</dependency>
```

Alternatively you can download the associated project and run the command `mvn install` and it will download all the required dependencies according to the pom.xml file in the project. Once we have the project running, we ran the below commands to compile the maven project and to run direct runner, spark runner, flink runner respectively.

```
mvn compile exec:java -Dexec.mainClass=TweetPipeline -Pdirect-runner
```

```
mvn compile exec:java -Dexec.mainClass=TweetPipeline -Pspark-runner
```

```
mvn compile exec:java -Dexec.mainClass=TweetPipeline -Pflink-runner
```

Following is similar command for running word count pipeline using the spark runner

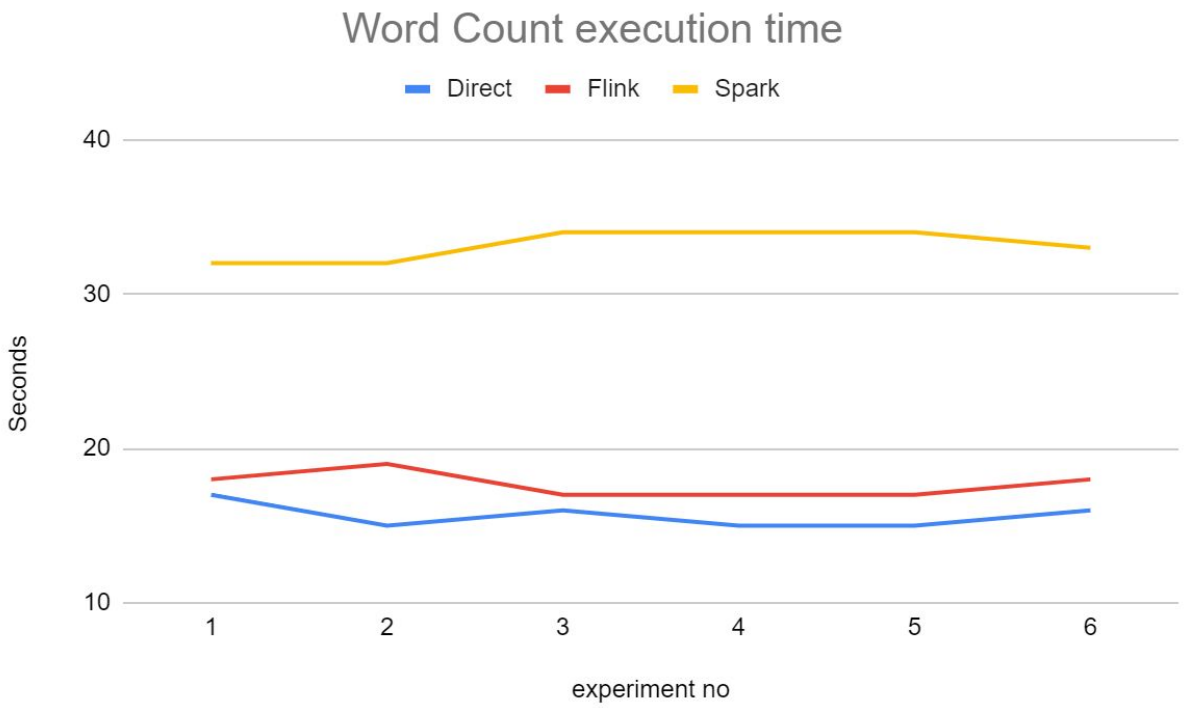
```
mvn compile exec:java -D exec.mainClass=org.apache.beam.examples.WordCount `
-D exec.args="--runner=SparkRunner --inputFile=pom.xml --output=counts" -P spark-runner
```

OBSERVATION

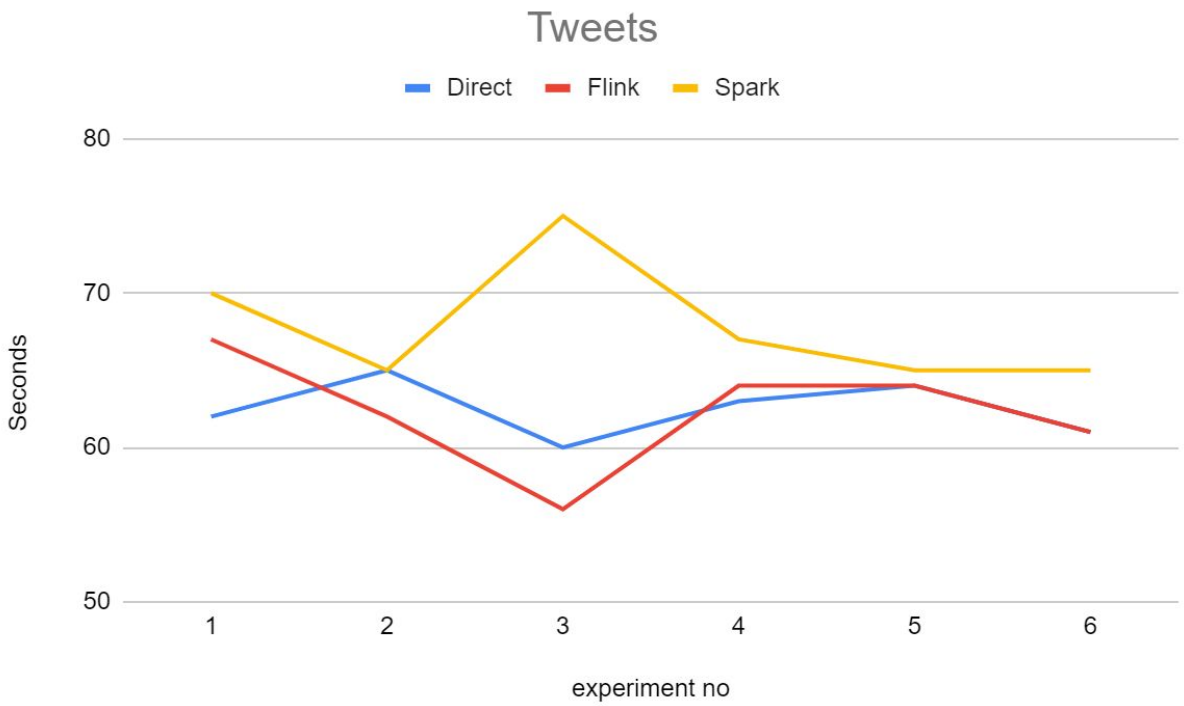
First run for each of the runner for beam had a higher value and was a huge outlier in the data so we discarded it. One of the possible reasons could be that it is taking time to initialize and download the dependencies. Another interesting observation was the continuous runs was

reducing the run time with each iteration until it stops decreasing and oscillate around a specific min value.

We haven't considered any specific correlation of actual execution time to data but it's correlation between different runners only. Word count example is reading Shakespeare's King Lear and applying the conventional map reduce program on it. Direct runner and Flink has almost the same performance with direct runner being ahead all the time. On the other hand, Spark is taking twice the time than other runners.



For the second experiment, we have developed the tweet pipeline. It reads the already collected tweets from JSON file, and identify the users who have tweeted more than 5 times. In this experiment, flink and direct runner has almost the same performance as in the previous experiment but surprisingly spark performed much better, even for one experiment it had the same performance as the direct runner.



CONCLUSION/ DISCUSSION

Apache beam is really amazing which helps us to write once and use multiple times on different platforms. We ran the Beam code for word count and tweet analysis on 3 different runners viz direct runner, Apache Flink and Apache Spark. All of these observations are independent from each other. We noticed that the direct outperforms others. When comparing Flink and Spark runner for the same code base, we got a better performance for Flink.

Additionally, the installations, project setup and execution for Beam was quite straightforward and the Beam documentation clearly mentioned them. But Apache beam API is still quite new and being developed as it is missing a lot features that are available in actually flink or spark framework. Few of the challenges that we faced during our project are listed below.

- Not good support for ML functions
- Not all of the functions aren't available in the Apache Beam
- Limited community support

- The only data structure which we could use for transformations was PCollection. But there was no way to extract any element from PCollection. The only possible solution was to write the output to a file and again read from file and continue processing.

Initially we used the cell tower data of Sweden which is provided by Opencellid (https://opencellid.org/) as our dataset. We tried to implement the K-Means algorithm to cluster the cellular tower data from Opencellid by creating the clusters of GSM and UMTS towers. But due to the above mentioned challenges, we were not able to do clustering using Beam. At the same time we were able to implement the K-means algorithm using the same dataset in Apache Flink. Hence, we ended up reporting results for simpler examples only but associated project contains non-functional implementation of K-means using Beam programming model. In the submission, we have 2 Java class Tweet.java and TweetPipeline.java which belongs to Tweet pipeline and rest of the code belongs to KMeans implementation.

FUTURE WORK

Although we ran experiments on different datasets but pipelines had similar DAG. We have potential to try it with more complex pipelines because each of the platform can have varying performance for different types of datasets. Furthermore, since all of these are distributed systems so it would be nice to evaluate the performance on the cloud or distributed infrastructure.

REFERENCES

1. https://medium.com/google-cloud/profiling-dataflow-pipelines-ddbbef07761d
2. https://stackoverflow.com/questions/50408933/does-apache-beam-support-iterative-algorithms-just-like-apache-flink
3. https://beam.apache.org/documentation/sdks/java/testing/nexmark/
4. https://github.com/apache/beam/blob/master/runners/google-cloud-dataflow-java/src/main/java/org/apache/beam/runners/dataflow/DataflowMetrics.java
5. https://www.youtube.com/watch?v=rJZEvp1pfCE
6. https://stackoverflow.com/questions/51274106/finding-the-key-with-maximum-value-in-a-pcollection-of-key-value-pairs/51274329
7. https://stackoverflow.com/questions/53235553/how-do-i-use-mapelements-and-kv-in-together-in-apache-beam
8. https://www.researchgate.net/post/Where_can_I_find_a_large_dataset_of_tweets
9. https://archive.org/search.php?query=collection%3Atwitterstream&sort=-publicdate